

# Optimized combinatorial clustering for stochastic processes

Jumi Kim<sup>1</sup> · Wookey Lee<sup>2</sup> · Justin Jongsu Song<sup>2</sup> · Soo-Bok Lee<sup>3</sup>

Received: 25 August 2016 / Accepted: 26 January 2017  
© The Author(s) 2017. This article is published with open access at Springerlink.com

**Abstract** As a new data processing era like Big Data, Cloud Computing, and Internet of Things approaches, the amount of data being collected in databases far exceeds the ability to reduce and analyze these data without the use of automated analysis techniques, data mining. As the importance of data mining has grown, one of the critical issues to emerge is how to scale data mining techniques to larger and complex databases so that it is particularly imperative for computationally intensive data mining tasks such as identifying natural clusters of instances. In this paper, we suggest an optimized combinatorial clustering algorithm for noisy performance which is essential for large data with random sampling. The algorithm outperforms conventional approaches through various numerical and qualitative thresholds like mean and standard deviation of accuracy and computation speed.

**Keywords** Nested partitions method · Optimized combinatorial clustering algorithm · Data clustering · Stochastic process

---

✉ Wookey Lee  
trinity@inha.edu

Jumi Kim  
jmkim@kosbi.re.kr

Justin Jongsu Song  
jaegal83@inha.edu

Soo-Bok Lee  
soobok@yonsei.ac.kr

<sup>1</sup> Korea Small Business Institute, 77 Sindaebang 1ga-gil, Dongjak-Ku, Seoul, Korea

<sup>2</sup> Department of Industrial Engineering, Inha University, Incheon, South Korea

<sup>3</sup> Department of Food and Enzyme Biotechnology, Yonsei University, Seoul, South Korea

## 1 Introduction

As a new data processing era like Big Data, Cloud Computing, and Internet of Things (IoT) approaches, the amount of data being collected in databases far exceeds the ability to reduce and analyze these data without the use of automated analysis techniques, data mining [25–27, 34, 36]. As the importance of data mining has grown, one of the critical issues to emerge is how to scale data mining techniques to larger and larger databases [2, 24, 35]. This is particularly true for computationally intensive data mining tasks such as identifying natural clusters of instances [10, 18]. Several approaches to scalability enhancements have been studied at length in the literature [4, 32], including using parallel mining algorithms [9, 23] and preprocessing the data by filtering out redundant or irrelevant features and thus reducing the dimensionality of the database [32]. Another approach to better scalability is using a selection of instance from a database rather than the entire database [29, 31, 41].

Perhaps the simplest approach to instance selection is random sampling [5, 6, 21]. Numerous authors have studied this approach for specific data mining tasks such as clustering [10, 18, 37, 38], association rule discovery [35], and decision tree induction [4]. When these approaches are implemented, one of the most challenging issues is determining a sample size that improves the performance of the algorithm without sacrificing the solution quality. Bounds can be developed that allow for a prediction of sample effort needed, but such bounds usually require knowing certain problem parameters and typically overestimate the necessary sample size [6, 14, 41]. On the other hand, too small sample will lead to a bias and degeneration in performance. One possible solution is to use adaptive sampling [4, 6, 24].

In this paper we advocate an alternative approach that is based on a novel formulation of the clustering task as

an optimization problem. We also take advantage of the fact that certain optimization techniques have been explicitly designed to account for noisy performance estimates, which are common when performance is estimated using simulation. In particular, one such method is the nested partitions method that can be used to solve general global optimization problems [32,39] and specifically combinatorial type optimization problems with noisy performance [19]. A characteristic of this method is that wrong moves made due to noise in performance estimates can be automatically corrected in a later move. In the scalable clustering context this means that noisy performance estimates, resulting from smaller samples of instances, may result in more steps taken by the algorithm but any bias will be automatically corrected. This eliminates the need to determine the exact sample size, although the computational performance of the algorithm may still depend on some extent on how it is selected.

Even though the pure NP method guarantees the convergence to the optimal solution, its efficiency and convergence properties can still be improved. To address these, two extensions to the pure NP method are suggested: the statistical selection method and random search method. First, to have more intelligent sampling, we use Nelson Matejcziks procedure [31]. Second, Genetic Algorithms (GAs) and k-means algorithm are used to speed convergence and to overcome the difficulty in the backtracking stage of the Nested Partitioning algorithm. For the numerical evaluation, two different types of cancer data are used. Using these extended algorithms, we want to show that the computation time can be reduced by sampling of the instances rather than using all the instances without affecting solution quality. Also we can give proper guideline for proper instances at least used.

**Organizations** The remainder of this paper is organized as follows. In Sect. 2 we briefly review statistical selection method and clustering techniques. In Sect. 3 we discuss the basis for the new clustering methodology, which is an optimization method called the Nested Partitions method and extended algorithm, Optimized Combinatorial NP Cluster algorithm. In Sect. 4 we present some numerical results of the scalability of the algorithm with respect to the instance dimension, and Sect. 5 contains concluding remarks and suggestions for future research directions.

## 2 Literature review

### 2.1 Statistical selection method and random search method

In the discrete event stochastic simulation, to choose the best solution is the maximum or minimum expected simulation result among a set of alternative solutions. Thus, Ranking and Selection (R&S) procedure is a primary matter of appre-

hension [7]. Bechhofer proposed the fundamentals of R&S at first [1]. The suggested original indifference zone R&S procedure [1] is a single-stage and presumes unknown means and known, common variances for all results. But it doesn't have to be single-stage. We can extend to multi-stage procedures (sequential procedures) assuming common, known variances by defining the user-specified number of observations. Bechhofer et al. [1] presented such methodologies and Koeing and Law [22] extended the indifference zone approach for sifting procedure. As contrasted with the articles discussed, Frey and Dueck [10] presented a representative exemplar procedure not requiring reduction to a univariate model. To allocate additional replications, the indifference zone procedures used a least-favorable configuration where the optimal computing budget allocation and Bayesian decision-theoretic methods used an average case analysis [5,8,33]. All three procedures are applicable to both two-stage and sequential procedures. These assume that simulation result is independent and normally distributed having unknown mean and variance.

Inoue et al. [16] showed empirically that the two-stage procedure [1,6] performs competitively with sequential optimal computing budget allocation model and Bayesian decision-theoretic methods when the number of systems under consideration is small ( $k < 5$ ). For a large number of systems ( $k \geq 5$ ), or when the difference in the mean output of the best system and other systems varies significantly, the two-stage procedure [1,6] is less effective at identifying the best system. Among two-stage procedures, the Bayesian decision-theoretic procedures have the best overall performance characteristics.

Recently, many articles have tried to unify the fields of R&S and multiple comparison procedures (MCPs). Multiple comparisons with the best (MCB) [31] is one of the most widely used MCPs. To apply MCB in a discrete-event simulation, the simulation runs must be independently seeded and the simulation output must be normally distributed, or averaged so that the estimators used are somewhat normally distributed [31,36,38]. There are four R&S-MCB procedures having normally distributed data, but do not require known or equal variance: Nelson and Matejcziks Procedure (Procedure NM) [31], two-stage procedure (Procedure B) [1,6], Watanabe (Procedure W) [40], and Frey and Dueck Procedure (Procedure FD) [10]. Procedure B and Procedure FD are performed in the same manner with the only difference being in the calculation of the sample means. Both algorithms require independence among all observations. The total sample size depends on the sample variance of the systems. So the larger the sample variance, the more replications are required. Unlike these algorithms, Procedure NM requires fewer total observations by employing the common random number where Watanabe [40] used the Bonferroni correction to account for the dependence induced by common random number. However, Nelson and Matejczik [31]

observed that the benefit gained from using Procedure W was diminished when the number of systems to be compared would be large. To overcome this problem, they presented Procedure NM where it assumed that the unknown variance covariance matrix exhibited a structure known as sphericity that implied the variances of all paired differences across systems were equal, even though the marginal variances and covariance may be unequal. The difference between Procedure W and NM is the calculation of sample variance. This sample variance affects the total number of sample size for second-stage sampling. Procedure B is superior to Procedure NM and DT in terms of the total observations required to obtain the desired confidence level. The only potential drawback with Procedure B is that the assumption of sphericity may not be satisfied.

When the feasible region is discrete, random search methods are generally used. These methods also cannot usually guarantee a global optimal, and therefore they are often called heuristics methods. Three common random search methods are mentioned below. Tabu Search was originally proposed by Glover [12] for escaping local optimal by using a list of prohibited solutions known as the tabu list. The commonly used diversification method is re-starting from the best solution obtained so far. Another drawback of the tabu search is unless there is a long tabu list, it may reach a previously visited solution. Simulated annealing (SA), introduced by Kirkpatrick et al. [20], is a random search method that is able to escape local optima using a probability function. Unlike the tabu search, SA does not evaluate the entire neighborhood in every iteration. Instead, it randomly chooses only one solution from the current neighborhood and evaluates its costs. That means SA tends to need more iterations to find the best solution than the tabu search method. Another disadvantage is that it does not have memory, and hence it may re-visit a recent solution. There is a combination method of tabu and SA. Genetic algorithms (GAs) were originally developed by Holland [15]. This is one of the most widely known evolutionary methods, which is both powerful and broadly applicable to stochastic optimization [41]. Commonly used operators include selection, reproduction, crossover, and mutation. It mimics the mechanisms of natural selection and natural genetics where stronger individuals are more likely to survive in a competing environment. Thereby, the strongest individual (having the best performance) survives.

## 2.2 Scalable clustering

Clustering has been an active area of research for several decades, and many clustering algorithms have been proposed in the literature [10, 11, 13, 28, 30]. In particular, considerable research has been devoted specifically to scalable clustering. We will start by briefly describing the various types of clus-

tering algorithms and then mention some specific scalable methods.

Clustering algorithms can be roughly divided into two categories: hierarchical clustering and partitional clustering [18]. In hierarchical clustering all of the instances are organized into a hierarchy that describes the degree of similarity between those instances (e.g., a dendrogram). Such representation may provide a great deal of information, but the scalability of this approach is questionable as the number of instances grows. Partitional clustering, on the other hand, simply creates one partition of the data where each instance falls into one cluster. Thus, less information is obtained but the ability to deal with a large number of instances is improved. Examples of the partitioning approach are the classic  $k$ -means and  $k$ -medoids clustering algorithms.

There are many other characteristics of clustering algorithms that must be considered to ensure scalability of the approach. For instance, most clustering algorithms are polythetic which means all features are considered simultaneously in tasks so as to determine the similarity of two instances. But if we have big features, this may pose scalability problems. For this reason, monothetic clustering algorithms that consider one feature at a time is considered. Most clustering algorithms are also non-incremental in the sense that all of the instances are considered simultaneously. However, there are a few algorithms that are incremental, which implies that they consider each instance separately. Such algorithms are particularly useful when the number of instances is large. Scalable clustering has received considerable attention in recent years, and here we will mention only a few of the methods. In the early stage, Guha et al., [13] presented the steps of the CURE algorithm that they obtained a sample from the original database, partition the sample into a set of partitions and then cluster each partition, eliminate outliers and cluster the partial clusters. Finally, each data instance is labeled with the corresponding cluster.

Such as  $k$ -means and  $k$ -medoids, improved scalable versions of partitioning methods. The Clustering LARge Applications (CLARA) algorithm improves the scalability of the PAM  $k$ -medoids algorithm by applying PAM to multiple samples of the actual data and returns the best clustering [18]. Jain and Dubes [17] suggest a single pass  $k$ -means clustering algorithm with the main idea to use a buffer to save points from the database in a compressed form. This approach was simplified by Farnstrom et al., [7] in an effort to reduce the overhead that otherwise might cancel out any scalability improvements that might be achieved.

There is another way of improving scalability via distributed clustering, where instead of combining all data before clustering, data sets are operated by the affinity propagation (AP) clustering algorithms [10]. The propose algorithm in this paper is a partitional clustering algorithm that try to find cluster centers and uses random sampling to

improve scalability. In that sense, it is the most similar to AP clustering algorithm, except it guarantees the s optimization solution.

### 3 Hybrid clustering algorithm

#### 3.1 Nested partitions (NP) method

The main framework of suggested algorithm was suggested by Shi and Iafsson [39] which is called Nested Partitions (NP) Method. This is an optimization method that solve general global optimization problems of the following form:

$$\min_{x \in X} f(x) \quad (1)$$

where  $x$  is a point in a  $n$ -dimensional space  $X$  and  $f : X \rightarrow R$  is a real-valued performance measure defined on this space. This performance may or may not be known deterministically. In our paper, we define  $X$  as the space of all clusters and the function measures some quality of the clusters.

The concept of the NP method is very simple. In each step, the method partitions the feasible region into subsets by the rule and concentrates on the computational effort in those subsets that are considered hopeful which might have the best answer. The partitioning rule depends on a case. Thus there is no fixed rule but that all subsets are disjoint. At each iteration of the algorithm.

We assume that there is a region, which is considered the most promising region having the most likely to contain the best solution at every iteration. Then this most promising region is partitioned into  $M$  regions and the remaining of the feasible region is aggregated into one region called the surrounding region. So we have  $M + 1$  disjoint subsets at each iteration. We sample using some random sampling scheme, and calculate an average of the performance function, a promising index for each of these  $M + 1$  regions. Then these promising indices are compared to determine which region has the most promising index, the smallest average of performance function. The best sub-region having the best performance is the most promising region. However, if the best performance is found in the surrounding region, the algorithm backtracks and a larger region containing the current most promising region becomes the new most promising region. We then partitioned and sampled in a similar fashion from this new most promising region. This process is repeated until the criteria for termination is satisfied. The main components of the method are:

- **Partitioning** At each iteration the feasible region is partitioned into subsets by predefined rule. This partitioning creates a tree of subsets which is called partitioning tree.

- **Creating feasible solutions** To evaluate each of the subsets, a randomly generated sample of solutions is obtained from each subset and used to estimate the performance of each region as a whole.
- **Evaluating promising index** To select the most promising region, calculate the promising index for each subregion.
- **Retracing** If the best solution is found in the surrounding region, the algorithm retraces to what was the most promising region in the previous iteration.

This method combines adaptable global sampling with local heuristic algorithm. It uses a supple partitioning method to divide the design space into regions. Each region should be evaluated individually and then aggregates the evaluation results from each region to determine the region for concentrating the computational effort. This means that the NP method intelligently samples from the entire design space and concentrates the sampling effort by methodical partitioning of the design space.

#### 3.2 Defining clusters

We can manage suggesting algorithm using the NP format. From the view of this approach we presume that we partition a whole data set into several and that each cluster is defined by its center (every instance are set to the nearest center). So the coordinates of each center of the cluster are the decision variables. We notate the  $j$ th cluster as  $x^{(j)} = (x_1^{(j)}, x_2^{(j)}, \dots, x_n^{(j)})$ , where  $j = 1, 2, \dots, m$ . Therefore, this clustering problem tries to locate the centers for optimizing certain performance.

In case of clustering defining a performance measure to be optimized is very nonobjective, because there are no standard criteria for constituting a good cluster. But we have the most common measures that can be used: probably maximizing similarity within a cluster (that is, maximizing homogeneity or compactness), and minimizing similarity between different clusters (that is, maximizing separability between the clusters).

A particular strength of suggesting algorithm is that it can adopt any measure of cluster performance, even combination of measures. We define the function  $f$  as the measure of the quality of a cluster. For performance comparison, we will compare our approach to other well-known methods that focus on the within similarity or compactness of a cluster. To make sure of performance comparison, we simplify the measure as a single measure of similarity within cluster:

$$f(x^{(1)}, x^{(2)}, \dots, x^{(m)}) = \sum_{y \in \psi} \sum_{i=1}^n |y_i - x_i^{[y]}|^2 \quad (2)$$

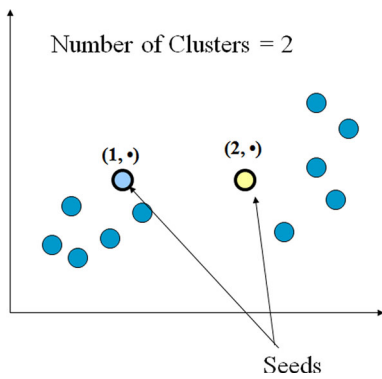


Fig. 1 Simple example for clustering using NP methodology

We define  $\psi$  as the space of all instances,  $y \in \psi$  as a specific instance in this space,  $x_i^{[y]}$  as the cluster center which the instance is assigned, and  $|y_i - x_i^{[y]}|$  as the difference between the  $i$ th coordinate of the instance and the homologous center. So the objective function is the sum of the distance of  $n$  data points from their respective cluster centers. By using such a simple measure we focus on the performance of the algorithm itself as we mentioned before.

When we use NP method, the main implementation issue is defining the partitioning rule. By finding cluster centers for one feature at a time, we manage this issue. In other words, at each level of the partitioning tree, the values for all centers are limited to a range for one feature. This bounds the subsets that make the partitioning tree. Using the idea of generic NP method, we do random sampling from each subset, and apply the  $k$ -means algorithm to those random samples to speed convergence. The resulting improved centers are used to select the most promising region at the next iteration. This most promising region is partitioned further and remaining regions are aggregated as surrounding region, and so forth.

Figure 1 shows that the simple example when we apply NP methodology to the clustering problem. This is the problem with two dimensions and the total number of cluster is 2. To simplify the problem, we assume that each dimension has only two values. That is,  $x_i = \{1, 2\}$ ,  $i = 1, 2$ . Figures 2 and 3 demonstrates a partitioning tree whereas all features can take two different values. The objective of this problem is to find the optimal location of 2 clusters (identified as  $C_1$  and  $C_2$ ). This partitioning method helps the scalability of the method with respect to the feature dimension. It focuses on fixing one feature at a time and repeated until all features are fixed at every iterations. During the random sampling stage, all features are used simultaneously to select subregions. This approach can thus be thought of as having elements of both monothetic and polythetic clustering. This partitioning approach helps the scalability of the method with respect to the feature dimension. It concentrates one feature at a time and is in that sense monothetic. But all features are

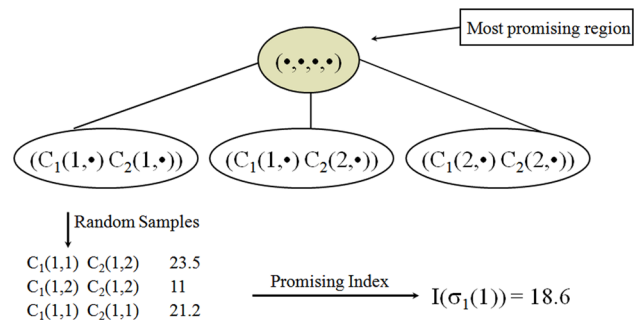


Fig. 2 First iteration of the example

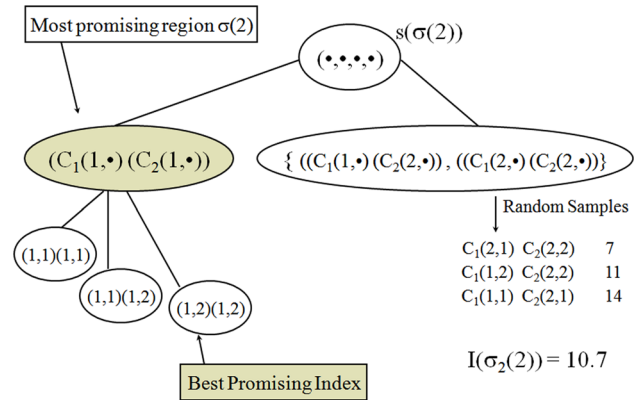


Fig. 3 Second iteration of the example

randomly assigned values during the random sampling stage, and thus all features are used simultaneously to select subregions. So this approach can be thought of both monothetic and polythetic clustering.

It is also significant to note that the partitioning tree makes a structure on the space of all possible clusters, and determines the effectiveness of the search through this space. Furthermore, investigating effective algorithms about ordering features is an important research topic in the future.

Figure 2 shows initial partitioning. First dimension of each cluster are set as (1,1) in the first subset. For second subset, first dimension of each cluster are set as (1,2) and for third subset, first dimension of each cluster are set as (2,2).

Random sampling is performed after partitioning of three subsets. For instance, every sample point of this subset has a fixed first dimension for the first subset; the first cluster and the second cluster are fixed as 1. Centers can be randomly assigned from the values 1,2 for the remaining dimension. A similarity value is calculated by formulation (2) using the sampling from each subset.

The promising index is calculated for each subset based on these values. The most promising region is the first subset having the smallest promising index after calculating promising index for all subsets.

Figure 3 shows that the most promising index is first subset at first iteration. The partitioning starts from the first subset of the second iteration. At the same way, the second dimension can take 2 different values, and three different subsets can be obtained like in the 1st iteration. The second iteration is the maximum depth because there are two dimensions in this problem. There is one more subset which is called the surrounding region from the second iteration. The subset which contains center  $C_1(1, \cdot)$ ,  $C_2(2, \cdot)$  is surrounding region (in Fig. 3). After sampling from all subsets, the most promising index is found in the second region, having the first clusters coordinate (1,1) and the second clusters coordinate (1,2). These coordinates are optimal because they minimize the similarity of the problem.

### 3.3 Optimized combinatorial cluster algorithms

As already mentioned, the NP method has two apparent drawbacks. There are two types of error in the estimate of each region: First, sampling error due to the use of sample points in the region, and the estimation error because of the use of simulation. Secondly, there is no guarantee whether the movement is correctly made in each iterations. To get over this problem, a two-stage method is suggested both of these problems [19, 27, 36]. It is possible to guarantee that the correct move is made by using statistical selection methods. Because statistical selection methods determine a second-stage sample size to use different numbers of sample points in each region, while simultaneously controlling the total error. To take on this hypothesis, we use Nelson and Matejcek [31] and incorporate with our scheme.

One main idea of statistical selection methods is that the number of sample points gained from each system should be proportional to the variance of the performance of each system. This method is very helpful when incorporated with NP scheme, especially the surrounding region which is expected having high variance than the other sub-regions needing a larger sampling size.

To state the two-stage approach rigorously, let  $D_{ij}(k)$  be the  $i$ th set of random sample points selected from the region  $\sigma_j(k)$  in the  $k$ th iteration, where  $i \geq 1$  and  $j = 1, 2, \dots, M + 1$ . In addition,  $N = |D_{ij}(k)|$ ,  $\theta \in D_{ij}(k)$ , and  $L(\theta)$  set as the initial number of sample points assuming constant, a point in that set and a simulation estimate of the performance of this point each. Then in the  $k$ th iteration, for every  $i$ ,

$$X_{ij}(k) = \min_{\theta \in D_{ij}(k)} L(\theta) \quad (3)$$

is an performance estimate of the region  $\sigma_j$ , which is referred as the  $i$ th system performance for the  $j$ th system,  $i \geq 1$ ,  $j = 1, 2, \dots, M + 1$ .

First, two-stage ranking and selection method acquires  $n_0$  system estimates. Then determines the total number of  $N_j$  of

system estimates using that information needed from the  $j$ th system, which is, subregion  $\sigma_j(k)$  based on the variance. If we want to choose the subregion as correctly with probability at least  $P^*$ , this number should be selected to be enough large to an indifference zone of  $\epsilon > 0$ .

First-stage samples are randomly obtained from each region by using random numbers for each region. Given a fixed first-stage sample size  $n_0$ , we can determine sample variance  $S$  of the difference of the sample means. Using final sample size given indifference zone  $\epsilon$  can be computed.

$$N = \max \left\{ n_0 \left[ \left( \frac{gS}{\epsilon} \right)^2 \right] \right\} \quad (4)$$

Note that this requires the constant  $g$  which affected by the initial sample size  $n_0$  and the number of regions  $M$  that are compared.

Genetic Algorithm (GA) is used since GA is one of the well-known and effective heuristic algorithms although there is no guarantee of global convergence. Therefore, sample points that better represent the best performance in their region can be obtained by applying GA search to each sub-region. The next promising region can be more exactly determined based on these sample points. This is because GA guarantees local optimums at least for each region by finding the best solution of each region. To improve the performance of the NP method, we combine the well-known heuristic clustering algorithm,  $k$ -means algorithm, also. As a result, a combined algorithm retains the benefits of all of the methods.

### 3.4 Three cluster algorithms

In this section, we suggest 3 types of Cluster Algorithms, Algorithm NP/NM/Km, Algorithm NP/NM/Genetic, and Algorithm NP/NM/Km/Genetic. To present a detail description in Table 1, we need the following notations:

The squared error criterion function is used as a performance measure. Its calculation is as follows.

**Table 1** The notations

Symbol	Description
$\Theta$	The feasible region
$\sigma(k)$	The most promising region in the $k$ th iteration
$s(\sigma)$	The super-region of $\sigma \subseteq X$
$d^*$	Maximum depth
$m$	Total number of clusters (given)
$n$	Total number of features (given)
$n_0$	The number of samples (given) of each subregion
$M_{\sigma(k)}$	The number of subregion at $k$ th iteration

$$J(z) = \sum_{i=1}^{N_C} \sum_{x \in I_j^i} |x - z_j^i|, i = 1, \dots, N, j = 1, 2, \dots, M_{\sigma(k)+1}$$

Using a sample of instances, the estimate

$$\hat{L}(z) = \sum_{i=1}^{N_C} \sum_{x \in I_j^i} |x - z_j^i|, i = 1, \dots, N, j = 1, 2, \dots, M_{\sigma(k)+1}$$

is used instead of  $J(z)$ .

We can now state the detailed algorithm.

## 4 Numerical results

In order to evaluate the performance of these algorithms, two different sizes of cancer data B-type and S-type are considered [3]. The B-type data set has 9 features and 699 instances; whereas, the S-type data set has 9 features and 286 instances. By varying the number of instances, we show that the algorithm can use a random sample of instances without sacrificing solution quality and determine appropriate guidelines for how many instances are needed. We vary the number of instances as 100, 50, 28, 15, 4.5, 1.5, 0.7, and 0.5% of total instances. In case of B-type data set, they are 699, 350, 200, 100, 50, 30, 10, 5, and 3. For S-type data set, we use 286, 143, 82, 41, 20, 12, 4, 2, and 1 as instances. Figure 4 shows numerical results of B-type cancer data set of each duplication. We can't find the pattern of similarity of this Fig. 4 (left). But we can notice that partial instances well performs in terms of computation time. In NP/NM/Km algorithm, the computation time (right) of 50% of instances is almost 1/4 of 100% of instances. From all algorithms, we find that using partial instances needs less computation time than using full instances. We can get similar results from S-type cancer data set in Fig. 5. We can save the time by hiring partial instances.

Table 2 shows that the mean and the standard deviation of accuracy, and computation speed for B-type and S-type data set. Let's see the B-type data set first. From these results, we get several important things. For every algorithm, we get the best solution in terms of solution quality when we use half instances. We can decrease the computation speed without changing the solution quality by using half of the instances. For example, in NP/NM/Km algorithm, the similarity value and computation time are 4259 and 394,780 each when all instances are used and 4208 and 101,670 each when 50% of instances are used. Computation time is cut almost 75% with no reduction in quality. We can get similar results from other algorithms. In NP/NM/Genetic algorithm, we can save time 63% when we use half instances. In NP/NM/Genetic/Km algorithm, we can get the best solution when we use 28% of

### Algorithm 1 NP/NM/Km algorithm

#### Step 1. Initialization

- 1: Set  $k = 0$  and  $\sigma(k) = \emptyset$ .
- 2: Specify the value of  $z_j^0, j = 1, 2, \dots, M_{\sigma(k)}$
- 3: Specify the constants  $\varepsilon, \alpha, n_k$  and  $n_0$ .
- 4: Let  $g = T_{k-1, (k-1)(n_0-1), 0.5}^{(\alpha)}$ , an equi-coordinate critical point of the equi-correlated multivariate central  $t$ -distribution

#### Step 2. Partitioning

- 5: If  $d(\sigma(k)) \neq d^*$ , that is,  $\sigma(k) \neq \Sigma_0$ , partition the fittest region,  $\sigma(k)$ , into  $M_{\sigma(k)}$  sub-regions  $\sigma_1(k), \dots, \sigma_{M_{\sigma(k)}}(k)$
- 6: If  $d(\sigma(k)) \neq 0$ , that is,  $\sigma(k) \neq \emptyset$ , aggregate the surrounding region  $\emptyset \setminus \sigma(k)$  into one region  $\sigma_{M_{\sigma(k)}}(k)$

#### Step 3. Sampling ▷ Stage I Sampling

- 7: Set  $t = 0$
- 8: Km algorithm,  $h = 1$
- 9: Randomly Assign Instance to the Clusters
- 10: Use random sampling to obtain  $N$  instances and assign to the centers for each of the regions  $\sigma_j(k), j = 1, 2, \dots, M_{\sigma(k)} + 1$
- 11: Calculate the Squared Error Criterion Function  

$$\hat{L}_h(\sigma_j(k)) < \sum_{i=1}^{N_C} \sum_{x \in I_j^i} |x - z_j^h|, i = 1, \dots, N_C, j = 1, 2, \dots, M_{\sigma(k)} + 1$$
 If  $\hat{L}_h(\sigma_j(k)) < \hat{L}_{h-1}(\sigma_j(k))$  then  $X_{tj}(k) = \hat{L}_h(\sigma_j(k))$
- 12: if  $h = n_k$  continue to line 14  
 otherwise let  $h = h + 1$  and go back to line 9
- 13: Change the Center of each Subregion  
 Change the centers of the value of the features  $> d(\sigma(k))$  for each cluster of each subregion and back to line 9
- 14: If  $t = n_0$  continue to Step 4  
 Otherwise let  $u = u + 1$  and go back to line 9

#### Step 4. Estimating Mean and Variance of First-State Sampling ▷ Stage II Sampling

- 15: Compute the approximate sample variance of the difference of the sample means  

$$S^2 = \frac{2 \sum_{j=1}^k \sum_{t=1}^{n_0} (X_{tj} - \bar{X}_t - \bar{X}_j + \bar{X}_{..})^2}{(k-1)(n_0-1)}$$
 where  $\bar{X}_t = \sum_{j=1}^k X_{tj}/k, \bar{X}_j = \sum_{t=1}^{n_0} X_{tj}/n_0$ , and  
 $\bar{X}_{..} = \sum_{t=1}^{n_0} \sum_{j=1}^k X_{tj}/kn_0$

#### Step 5. Computing Total Sample Size for Second-Stage Sampling

- 16: Compute the total sample size for all  $j$   
 Compute the total sample size  $N_j(k) = \max \left\{ n_0, \left\lceil \left( \frac{gS}{\varepsilon} \right)^2 \right\rceil \right\}$   
 for  $j = 1, 2, \dots, M_{\sigma(k)+1}$

#### Step 6. Second-Stage Sampling

- 17: Obtain  $N_j(k) - n_0$  more sample estimates of the system performance for all  $j$  as in Step 3 above

#### Step 7. Estimating Mean of Second-Stage Sampling

- 18: Let the overall sample mean be the promising index for all  $j \in I$ ,  

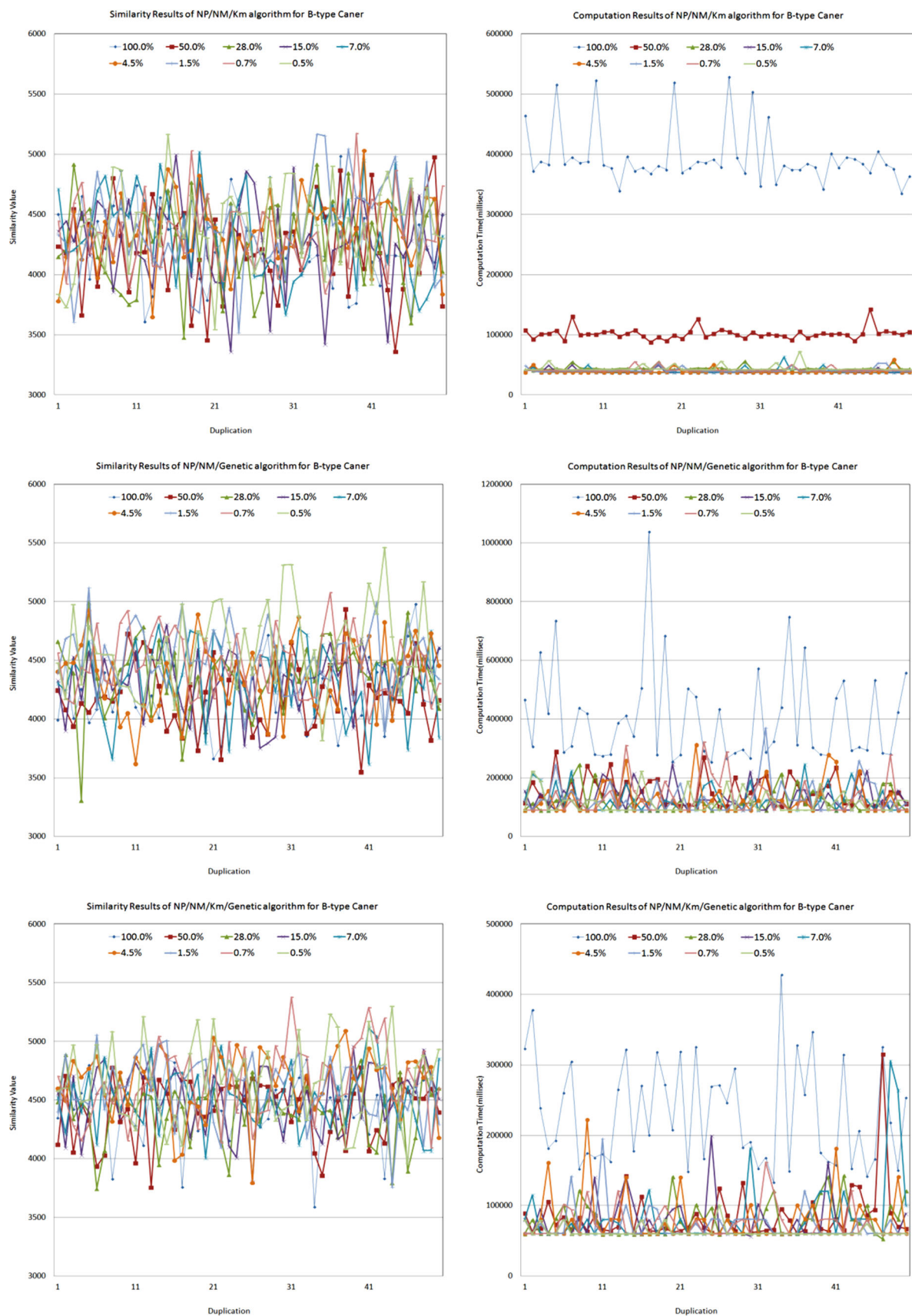
$$\hat{I}(\sigma_j(k)) = \bar{X}_j(k) = \frac{\sum_{t=1}^{N_j(k)} X_{tj}(k)}{N_j(k)}$$

#### Step 8. Calculating the Promising Index

- 19: Calculate the index of the region with the smallest squared error criterion function (most promising region)  
 $\hat{j}^k \in \arg \min \hat{I}(\sigma_j)$  for all  $j \in I$
- 20: If more than one region is equally promising, the tie can be broken arbitrarily. If this index corresponds to a region that is a sub-region of  $\sigma(k)$ , then let this sub-region be the most promising region of next iteration, that is,  
 $\sigma(k+1) = \sigma_j(k), j < M_{\sigma(k)}$
- 21: Otherwise, if the index corresponds to the surrounding region, backtrack to the region,  $s(\sigma(k))$ , of the current most promising region
- 22: That is, let  $\sigma(k+1) = s(\sigma(k))$

#### Step 9. Checking the Stopping Rule

- 23: If  $\sigma(k+1) \in n_0$  stop and  $\sigma_{opt} = \sigma(k+1)$  else  $k = k + 1$  and go back to Step 2



**Fig. 4** Numerical results of B-type cancer data set (left similarity value, right computation time)



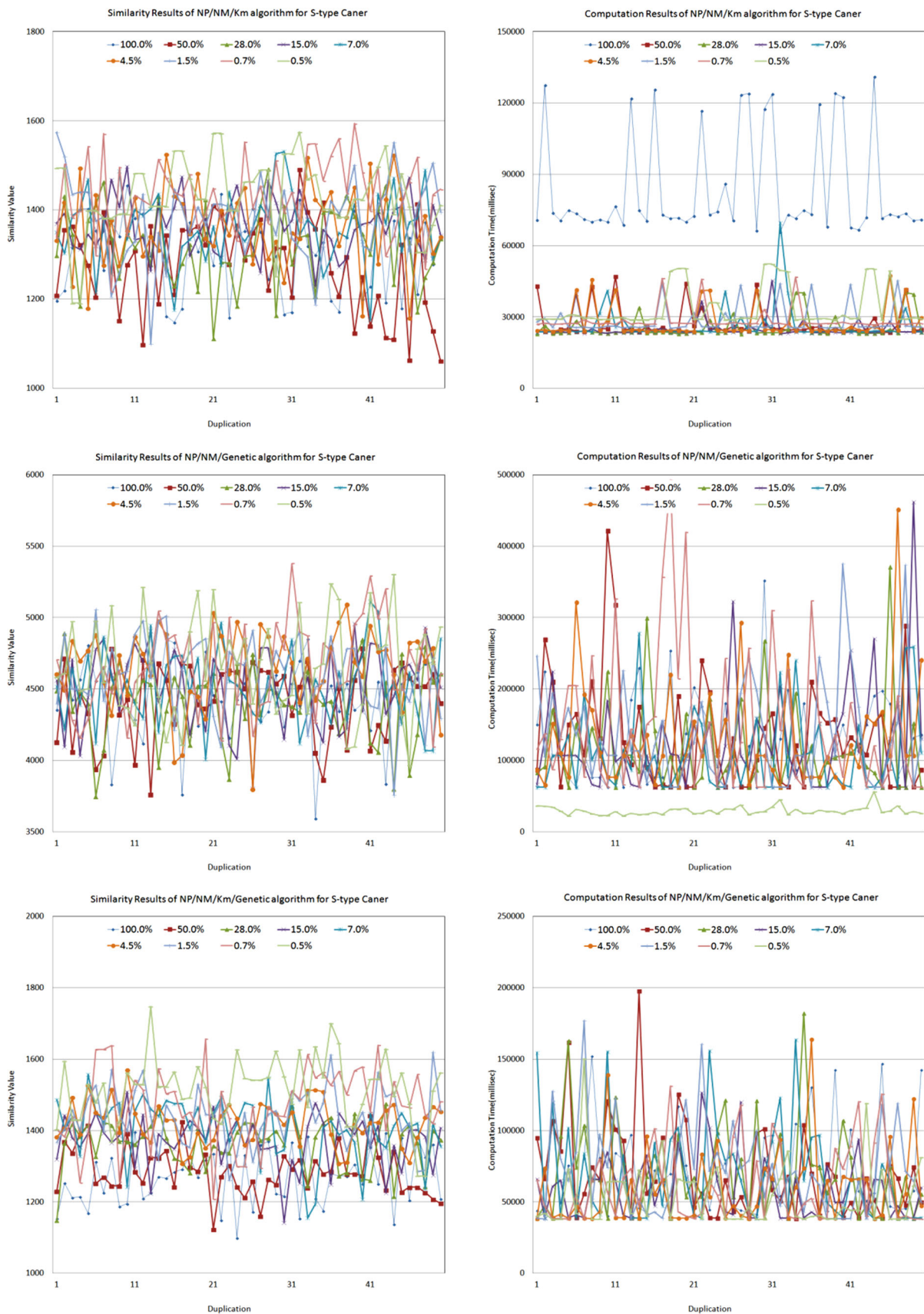


Fig. 5 Numerical results of S-type cancer data set

**Table 2** Effect of using fraction of instance space for the B-type and S-type data sets

Fraction no. of instances	B-type data set		S-type data set	
	Similarity value Mean $\pm$ SD	Computation time Mean $\pm$ SD	Similarity value Mean $\pm$ SD	Computation time Mean $\pm$ SD
<b>NP/NM/Km</b>				
100% (699)	4259.0 $\pm$ 46	3,94,777 $\pm$ 6668	1302.3 $\pm$ 13	84,115 $\pm$ 3166
50% (350)	4207.7 $\pm$ 53	101,666 $\pm$ 1352	1276.6 $\pm$ 15	27,295 $\pm$ 901
28% (200)	4264.8 $\pm$ 51	43,794 $\pm$ 498	1322.9 $\pm$ 12	25,700 $\pm$ 689
15% (100)	4272.9 $\pm$ 53	38,374 $\pm$ 498	1364.3 $\pm$ 9	25,160 $\pm$ 604
7% (50)	4331.4 $\pm$ 50	38,826 $\pm$ 660	1360.6 $\pm$ 11	26,603 $\pm$ 1074
4.5% (30)	4363.4 $\pm$ 41	38,966 $\pm$ 590	1363.1 $\pm$ 13	27,698 $\pm$ 960
1.5% (10)	4380.0 $\pm$ 53	40,622 $\pm$ 534	1381.7 $\pm$ 13	28,468 $\pm$ 850
0.7% (5)	4337.6 $\pm$ 43	41,641 $\pm$ 481	1434.8 $\pm$ 12	28,545 $\pm$ 656
0.5% (3)	4401.1 $\pm$ 49	44,065 $\pm$ 775	1430.9 $\pm$ 12	33,774 $\pm$ 1203
<b>NP/NM/Genetic</b>				
100% (699)	4203.7 $\pm$ 40	404,534 $\pm$ 23,118	1317.1 $\pm$ 12	128,158 $\pm$ 8684
50% (350)	4198.7 $\pm$ 41	152,181 $\pm$ 7237	1277.7 $\pm$ 11	134,849 $\pm$ 10,827
28% (200)	4369.5 $\pm$ 43	120174 $\pm$ 5522	1367.6 $\pm$ 11	116,872 $\pm$ 9181
15% (100)	4310.7 $\pm$ 38	127,696 $\pm$ 5884	1365.7 $\pm$ 10	128098 $\pm$ 10647
7% (50)	4317.5 $\pm$ 45	124,951 $\pm$ 6285	1397.4 $\pm$ 12	110209 $\pm$ 8235
4.5% (30)	4362.2 $\pm$ 44	129,202 $\pm$ 7907	1416.1 $\pm$ 11	130,608 $\pm$ 10,622
1.5% (10)	4518.5 $\pm$ 38	130,634 $\pm$ 7752	1430.1 $\pm$ 11	137,483 $\pm$ 10495
0.7% (5)	4466.8 $\pm$ 38	130,801 $\pm$ 8634	1435.8 $\pm$ 14	152,132 $\pm$ 14361
0.5% (3)	4606.7 $\pm$ 51	115,850 $\pm$ 5352	1452.2 $\pm$ 12	114,601 $\pm$ 10317
<b>NP/NM/Genetic/Km</b>				
100% (699)	4444.2 $\pm$ 39	231,200 $\pm$ 10,487	1267.8 $\pm$ 11	72,739 $\pm$ 4528
50% (350)	4418.7 $\pm$ 37	84,920 $\pm$ 5610	1285.9 $\pm$ 9	70,753 $\pm$ 4701
28% (200)	4401.2 $\pm$ 38	80,075 $\pm$ 3527	1363.0 $\pm$ 10	70,175 $\pm$ 4589
15% (100)	4514.5 $\pm$ 39	77,117 $\pm$ 4131	1370.8 $\pm$ 10	58,568 $\pm$ 3102
7% (50)	4512.6 $\pm$ 37	84,622 $\pm$ 6775	1405.2 $\pm$ 12	69846 $\pm$ 5070
4.5% (30)	4652.4 $\pm$ 39	80,971 $\pm$ 4968	1421.0 $\pm$ 9	59,105 $\pm$ 3929
1.5% (10)	4600.0 $\pm$ 38	72,910 $\pm$ 3476	1437.5 $\pm$ 12	72,188 $\pm$ 4561
0.7% (5)	4664.5 $\pm$ 44	74,396 $\pm$ 3163	1471.5 $\pm$ 15	62,197 $\pm$ 3847
0.5% (3)	4661.6 $\pm$ 49	63,127 $\pm$ 1185	1516.4 $\pm$ 13	54,164 $\pm$ 3241

instances. Nevertheless, we can cut down the computation time by 35% of full instances. Similar results are acquired from S-type data set. The NP/NM/Km algorithm and NP/NM/Genetic algorithm show that 50% of instances give the best solution. The computation speed of NP/NM/Genetic algorithm does not make of much of difference. Unlike the above algorithms, NP/NM/Genetic/Km algorithm gives the best solution when full instances are used. At both B-type and S-type data set, we get same computation results. Computation time is the best when 0.5% of instances is used in NP/NM/Genetic algorithm and NP/NM/Genetic/Km algorithm and 15% of instances is the best when NP/NM/Km algorithm is used.

Figures 6 and 7 are the average of similarity and computation time for B-type and S-type data set. As we already commented, 50% of instances give the best similarity value regardless of type and algorithm. Generally, NP/NM/Km algorithm gives the best solution. And NP/NM/Genetic/Km algorithm gives the worst solution. This results definitely clear for B-type data set, a larger data. The more fractions we hire, the better the solution we get. Similar results happen in computation speed. The NP/NM/Km algorithm is the best and NP/NM/Genetic algorithm is the worst. And this is apparently clear at larger data set. Especially the patterns between the algorithms are same at larger data set. Computation time speedily cut at 50% fraction and not much

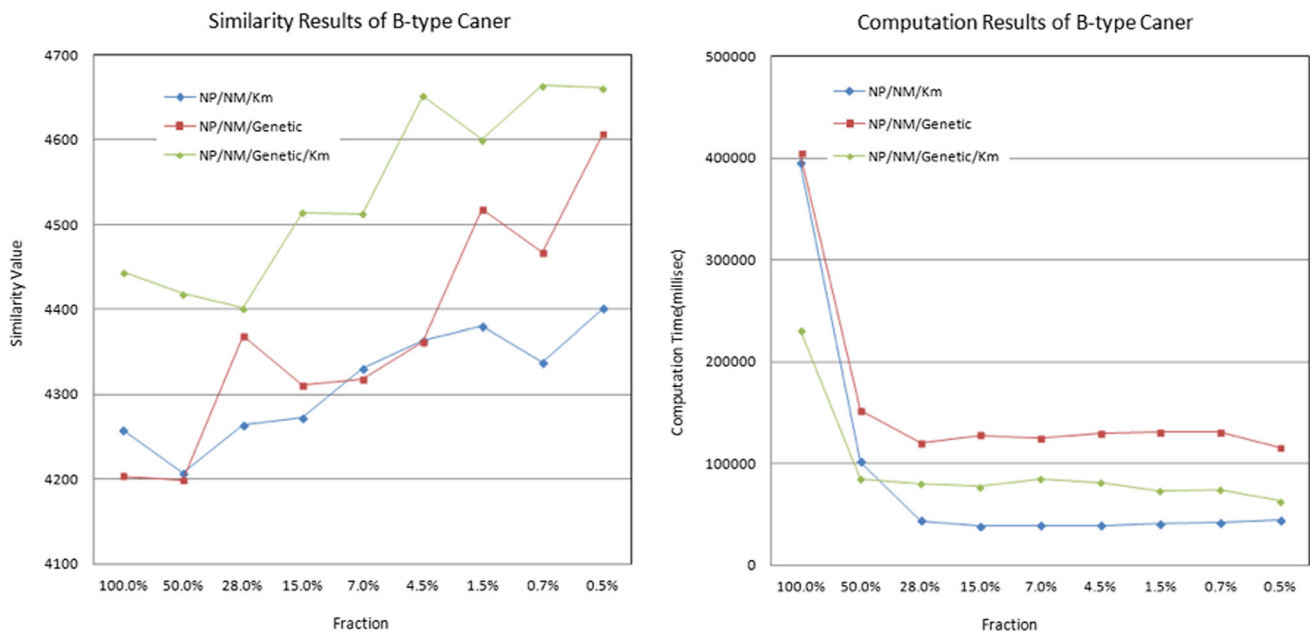


Fig. 6 Numerical results for different fraction of instances used for B-type cancer data set

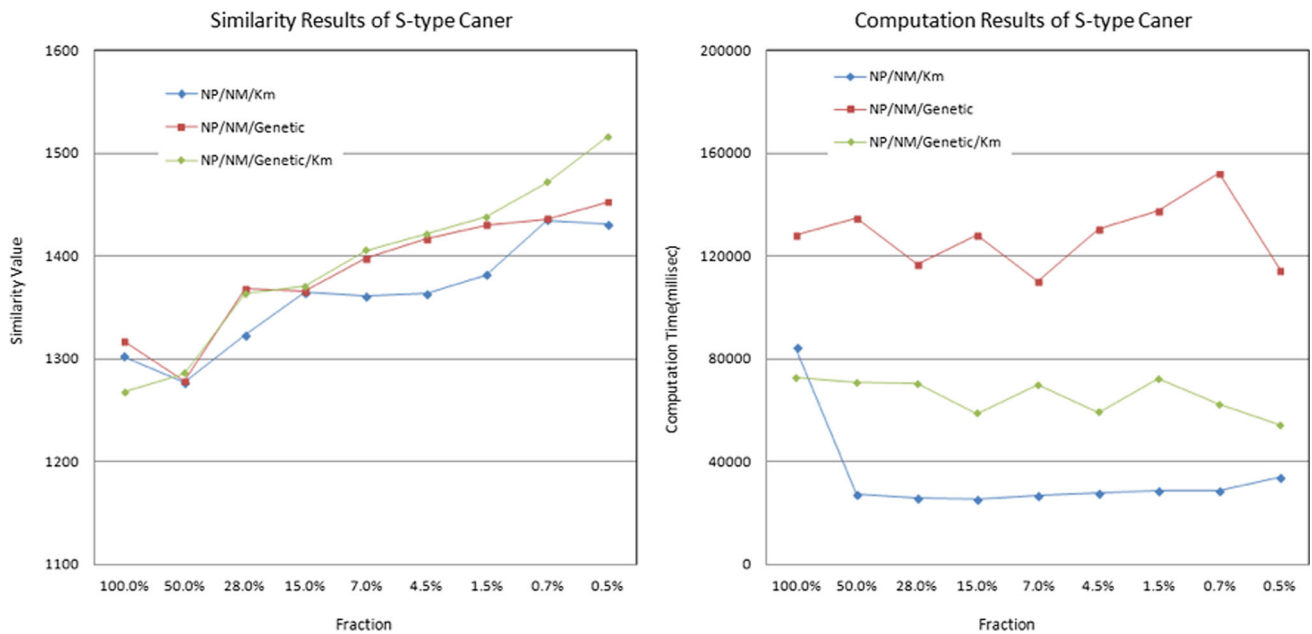


Fig. 7 Numerical results for different fraction of instances used for S-type cancer data set

difference of computation quality after 28% fraction which means if we have to choose between 25–0% fraction, we should choose 25% fraction.

### 5 Conclusions and future research

In this paper, we suggest Optimized Combinatorial NP Cluster algorithms for stochastic process that will be crucial for

B-type, complex data with random sampling. As we can see, the computation time can be cut by fraction of instances rather than using all the instances. This is more noticeable in cases of B-type data problem, larger data set. When only half of the instances are used, the computation time is cut without affecting solution quality. In addition, the standard deviation is declines, which means computation time is getting stable. But with too few instances, the solution quality becomes significantly worse while the computation time goes up. And

hiring  $k$ -means algorithm than Genetic algorithm gives better solution.

---

### Algorithm 2 NP/NM/Genetic algorithm

---

#### Step 1. Initialization

- 1: Set  $k = 0$  and  $\sigma(k) = \emptyset$ .
- 2: Specify the value of  $z_j^0$ ,  $j = 1, 2, \dots, M_{\sigma(k)}$
- 3: Specify the constants  $\varepsilon, \alpha, n_k$  and  $n_0$ .
- 4: Let  $g = T_{k-1, (k-1)(n_0-1), 0.5}^{(\alpha)}$ , an equi-coordinate critical point of the equi-correlated multivariate central  $t$ -distribution

#### Step 2. Partitioning

- 5: If  $d(\sigma(k)) \neq d^*$ , that is,  $\sigma(k) \neq \sum_0$ , partition the fittest region,  $\sigma(k)$ , into  $M_{\sigma(k)}$  sub-regions  $\sigma_1(k), \dots, \sigma_{M_{\sigma(k)}}(k)$
- 6: If  $d(\sigma(k)) \neq 0$ , that is,  $\sigma(k) \neq \emptyset$ , aggregate the surrounding region  $\emptyset \setminus \sigma(k)$  into one region  $\sigma_{M_{\sigma(k)}}(k)$

#### Step 3. Initial Population

- 7: If  $k = 0$  and  $d(\sigma(k)) \neq d^*$ , use random sampling to obtain an initial center population  $N$  strings from each of regions  $\sigma_j(k)$ ,  $j = 1, 2, \dots, M_{\sigma(k)} + 1$   
 $POP_j^j = [Z_j^1, Z_j^2, \dots, Z_j^N], j = 1, 2, \dots, M_{\sigma(k)} + 1$
- 8: Part of the lacks should be fulfilled using uniform sampling

#### Step 4. Sampling ▷ Stage I Sampling

- 9: Set  $t = 0$
- 10: GA Search  
 Apply the GA to each initial population  $POP_j^j$  individually, obtaining a final population for each region  $\sigma_j(k)$ ,  $j = 1, 2, \dots, M_{\sigma(k)} + 1$   
 $POP_F^j = [Z_F^1, Z_F^2, \dots, Z_F^N], j = 1, 2, \dots, M_{\sigma(k)} + 1$
- 11: Calculate the Squared Error Criterion Function (Overall Fitness)  
 Randomly assign instances to the best of final population to calculate squared error criterion function  
 $\hat{L}_h(\sigma_j(k)) = \sum_{i=1}^{N_C} \sum_{x \in I_j^i} |x - z_j^{Best}|$   
 $, i = 1, \dots, N_C, j = 1, 2, \dots, M_{\sigma(k)} + 1$   
 If  $\hat{L}_h(\sigma_j(k)) < \hat{L}_{h-1}(\sigma_j(k))$  then  $X_{ij}(k) = \hat{L}_h(\sigma_j(k))$
- 12: if  $h = n_0$  continue to step 5  
 Otherwise let  $h = h + 1$  and go back to line 10

#### Step 5. Estimating Mean and Variance of First-State Sampling ▷ Stage II Sampling

- 13: See Step 4 in Algorithm NP/NM/Km

#### Step 6. Computing Total Sample Size for Second-Stage Sampling

- 14: See Step 5 in Algorithm NP/NM/Km

#### Step 7. Second-Stage Sampling

- 15: Obtain  $N_j(k) - n_0$  more sample estimates of the system performance for all  $j$  as in Step 4 above

#### Step 8. Estimating Mean of Second-Stage Sampling

- 16: See Step 7 in Algorithm NP/NM/Km

#### Step 9. Calculating the Promising Index

- 17: See Step 8 in Algorithm NP/NM/Km

#### Step 10. Checking the Stopping Rule

- 18: See Step 9 in Algorithm NP/NM/Km
- 

---

### Algorithm 3 NP/NM/Km/Genetic algorithm

---

#### Step 1. Initialization

- 1: See Step 1 in Algorithm NP/NM/Genetic

#### Step 2. Partitioning

- 2: See Step 4 in Algorithm NP/NM/Genetic

#### Step 3. Initial Population

- 3: See Step 3 in Algorithm NP/NM/Genetic

#### Step 4. GA Search

- 4: Set  $t = 0$
- 5: GA Search  
 Apply the GA to each initial population  $POP_j^j$  individually, obtaining a final population for each region  $\sigma_j(k)$ ,  $j = 1, 2, \dots, M_{\sigma(k)} + 1$   
 $POP_F^j = [Z_F^1, Z_F^2, \dots, Z_F^N], j = 1, 2, \dots, M_{\sigma(k)} + 1$

#### Step 5. First-Stage Sampling

- 6: Set  $t = 0$
- 7: Km algorithm  
 See Step 3 in Algorithm NP/NM/Km

#### Step 6. Estimating Mean and Variance of First-State Sampling ▷ Stage II Sampling

- 8: See Step 4 in Algorithm NP/NM/Km

#### Step 7. Computing Total Sample Size for Second-Stage Sampling

- 9: See Step 5 in Algorithm NP/NM/Km

#### Step 8. Second-Stage Sampling

- 10: See Step 6 in Algorithm NP/NM/Km

#### Step 9. Estimating Mean of Second-Stage Sampling

- 11: See Step 7 in Algorithm NP/NM/Km

#### Step 10. Calculating the Promising Index

- 12: See Step 8 in Algorithm NP/NM/Km

#### Step 11. Checking the Stopping Rule

- 13: See Step 9 in Algorithm NP/NM/Km
- 

For further research, we can extend these algorithms to the more various statistical selection method and random search method that can be expected more quality in computation and similarity.

**Acknowledgements** This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MOE) (NRF-2016R1A2B4014245, NRF-2016R1E1A2915555) and Yonsei University.

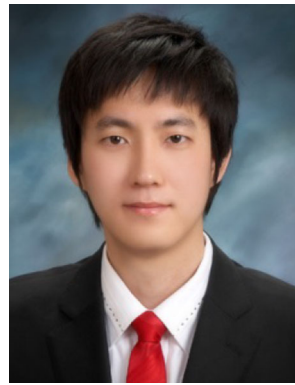
**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

## References

1. Bechhofer, R.E., Kiefer, J., Sobel, M.: Sequential Identification and Ranking Procedures: With Special Reference to Koopman-Darmois Populations, vol. 3. University of Chicago Press, Chicago (1968)
2. Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N.: Scheduled sampling for sequence prediction with recurrent neural networks. *Adv. Neural Inf. Process. Syst.* **28**, 1171–1179 (2015)
3. Blake, C., Merz, C.J.: {UCI} repository of machine learning databases (1998)
4. Chauchat, J.H., Rakotomalala, R.: Sampling strategy for building decision trees from very large databases comprising many continuous attributes. *Instance Selection and Construction for Data Mining*, pp. 171–188. Springer, Berlin (2001)
5. Chen, X., Ankenman, B., Nelson, B.L.: Common random numbers and stochastic kriging. In: *Proceedings of the Winter Simulation Conference*, pp. 947–956. Winter Simulation Conference (2010)
6. Chick, S.E., Frazier, P.: Sequential sampling with economics of selection procedures. *Manag. Sci.* **58**(3), 550–569 (2012)
7. Farnstrom, F., Lewis, J., Elkan, C.: Scalability for clustering algorithms revisited. *ACM SIGKDD Explor. Newsl.* **2**(1), 51–57 (2000)
8. Ferrari, D.G., De Castro, L.N.: Clustering algorithm selection by meta-learning systems: a new distance-based problem characterization and ranking combination methods. *Inf. Sci.* **301**, 181–194 (2015)
9. Forman, G., Zhang, B.: Distributed data clustering can be efficient and exact. *ACM SIGKDD Explor. Newsl.* **2**(2), 34–38 (2000)
10. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. *Science* **315**, 972–976 (2007)
11. Fu, X., Niu, Z., Yeh, M.K.: Research trends in sustainable operation: a bibliographic coupling clustering analysis from 1988 to 2016. *Cluster Comput.* **19**(4), 2211–2223 (2016)
12. Glover, F.: Heuristics for integer programming using surrogate constraints. *Decis. Sci.* **8**(1), 156–166 (1977)
13. Guha, S., Rastogi, R., Shim, K.: Cure: an efficient clustering algorithm for large databases. In: *ACM SIGMOD Record*, vol. 27, pp. 73–84. ACM (1998)
14. Gupta, S.S., Miescke, K.J.: Bayesian look ahead one-stage sampling allocations for selection of the best population. *J. Stat. Plan. Inference* **54**(2), 229–244 (1996)
15. Holland, J.H.: *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, Ann Arbor (1975)
16. Inoue, K., Chick, S.E., Chen, C.H.: An empirical evaluation of several methods to select the best system. *ACM Trans. Model. Comput. Simul. (TOMACS)* **9**(4), 381–407 (1999)
17. Jain, A.K., Dubes, R.C.: *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River (1988)
18. Kaufman, L., Rousseeuw, P.J.: *Finding groups in data: an introduction to cluster analysis*, vol. 344. Wiley, New York (2009)
19. Kim, J., Yang, J., Ólafsson, S.: An optimization approach to partitioned data clustering. *J. Oper. Res. Soc.* **60**(8), 1069–1084 (2009)
20. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. *Science* **220**(4598), 671–680 (1983)
21. Kivinen, J., Mannila, H.: The power of sampling in knowledge discovery. In: *Proceedings of the thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 77–85. ACM (1994)
22. Koenig, L.W., Law, A.M.: A procedure for selecting a subset of size  $m$  containing the 1 best of  $k$  independent normal populations, with applications to simulation. *Commun. Stat. Simul. Comput.* **14**(3), 719–734 (1985)
23. Kotyrba, M., Volná, E., Oplatková Komínková, Z.: Comparison of modern clustering algorithms for twodimensional data. In: *Proceedings-28th European Conference on Modelling and Simulation, ECMS 2014*. European Council for Modelling and Simulation (2014)
24. Kumar, S., Mohri, M., Talwalkar, A.: On sampling-based approximate spectral decomposition. In: *ICML'09*, pp. 553–560. ACM, New York, NY, USA (2009)
25. Lee, C.G., Lee, W.: Analysis of hollywood motion picture by dea and its application of classification system. *J. Inf. Technol. Arch.* **13**(3), 487–495 (2016)
26. Lee, W., Leung, C.K.S., Lee, J.J.: Mobile web navigation in digital ecosystems using rooted directed trees. *IEEE Trans. Ind. Electron.* **58**(6), 2154–2162 (2011)
27. Lee, W., Loh, W.K., Sohn, M.M.: Searching steiner trees for web graph query. *Comput. Ind. Eng.* **62**(3), 732–739 (2012)
28. Li, L., Ye, J., Deng, F., Xiong, S., Zhong, L.: A comparison study of clustering algorithms for microblog posts. *Cluster Comput.* **19**(3), 1333–1345 (2016)
29. Liu, T., Rosenberg, C., Rowley, H.A.: Clustering billions of images with large scale nearest neighbor search. In: *IEEE Workshop on Applications of Computer Vision, 2007. WACV'07*, pp. 28–28. IEEE (2007)
30. Llanes, A., Cecilia, J.M., Sánchez, A., García, J.M., Amos, M., Ujaldón, M.: Dynamic load balancing on heterogeneous clusters for parallel ant colony optimization. *Cluster Comput.* **19**(1), 1–11 (2016)
31. Nelson, B.L., Matejcek, F.J.: Using common random numbers for indifference-zone selection and multiple comparisons in simulation. *Manag. Sci.* **41**(12), 1935–1945 (1995)
32. Olafsson, S.: *Improving scalability of e-commerce systems with knowledge discovery*. Scalable Enterprise Systems, pp. 193–216. Springer, Berlin (2003)
33. Pan, W., Zhong, H., Xu, C., Ming, Z.: Adaptive bayesian personalized ranking for heterogeneous implicit feedbacks. *Knowl. Based Syst.* **73**, 173–180 (2015)
34. Reed, D.A., Dongarra, J.: Exascale computing and big data. *Commun. ACM* **58**(7), 56–68 (2015)
35. Riondato, M., Upfal, E.: Efficient discovery of association rules and frequent itemsets through sampling with tight performance guarantees. *ACM Trans. Knowl. Discov. Data* **8**(4), 20:1–20:32 (2014)
36. Robinson, S., Worthington, C., Burgess, N., Radnor, Z.J.: Facilitated modelling with discrete-event simulation: reality or myth? *Eur. J. Oper. Res.* **234**(1), 231–240 (2014)
37. Satuluri, V., Parthasarathy, S., Ruan, Y.: Local graph sparsification for scalable clustering. In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, pp. 721–732. ACM (2011)
38. Shams, I., Ajourlou, S., Yang, K.: Modeling clustered non-stationary poisson processes for stochastic simulation inputs. *Comput. Ind. Eng.* **64**(4), 1074–1083 (2013)
39. Shi, L., Ólafsson, S.: Nested partitions method for global optimization. *Oper. Res.* **48**(3), 390–407 (2000)
40. Watanabe, H., Hyodo, M., Seo, T., Pavlenko, T.: Asymptotic properties of the misclassification rates for Euclidean distance discriminant rule in high-dimensional data. *J. Multivar. Anal.* **140**, 234–244 (2015)
41. Whitley, D., Howe, A.E., Hains, D.: Greedy or not? Best improving versus first improving stochastic local search for maxsat. In: *AAAI. Citeseer* (2013)



**Jumi Kim** received Ph.D. degree in Industrial Engineering at Iowa State University, USA, in 2002. She is now Senior Research fellow at Korea Small Business Institute, Seoul, Republic of Korea. Her research areas are Optimization, Data Mining, SME's Start Up and R&D.



**Justin Jongsu Song** received his B.Sc. and M.Sc. in Industrial Engineering, with high honors, from Inha University, Korea, in 2012. He is currently in Ph.D. candidate in Inha University. His research interests include Graph theory, Social network, Information Retrieval, and patent analysis.



**Wookey Lee** received the B.S., M.S., and Ph.D. from Seoul National University, Korea, and the M.S.E. degree from Carnegie Mellon University, USA. He currently is a Professor in Inha University, Korea. He has served as chairs and PC members for many conferences such as CIKM, DASFAA, IEEE DEST, VLDB, BigComp, EDB, etc. He is currently one of the Executive Committee members of IEEE TCDE. He won the best paper awards in IEEE TCSC, KORMS

and KIISE. Now he is the EIC of Journal of Information Technology and Architecture, and an associate editor for WWW Journal. His research interests include Cyber-Physical systems, Graph and Mobile systems, Data Anonymization, and Patent Information.



**Soo-Bok Lee** received the agricultural degree on food science and technology from Kyoto University, Kyoto, Japan, in 1997 and received the Ph.D. degree in food science and technology at Kyoto University, Kyoto, Japan, in 1997. He is now a member of department of food and nutrition, Yonsei University, Seoul, Republic of Korea. His research interests include Food Biotechnology, Enzyme Engineering, Glycobiotechnology, Functional Carbohydrates,

Extremophilic Enzymes, Nucleotide Sugars, and Enzymatic Bioconversion.